A System to promote agenda for a better society

The ProLobby Company is seeking to build a new system – "PromoIt" that handles the social promotion of the non-profit organization and other related campaigns. The first version of the System – The Minimal Viable Product (MVP), is limited, but it can be extended in the future. For the first version, we only use the Twitter social network.

The system's goal is to promote social campaigns. The means to do so involves onboarding business organizations that donate products, onboarding non-profit organizations that want to promote campaigns, and onboarding social activists – users of Twitter that can promote those campaigns.

The system users:

1. ProLobby Owner – the company representative. This user manages the system.
2. Non-Profit Organization Representative – the user that creates a campaign
3. Business Company Representative – the user that represents a company that donates products for campaigns
4. Social Activists – Twitter users that promote campaigns

User Stories (Functional Requirements)

1. As a non-profit representative, I'd like to register to the system to be able to create a campaign
   1. I'll provide a name, email, and a link to my organization's Website
2. As a non-profit organization representative, I'd like to create a campaign
   1. I'll provide a link (URL) to the campaign landing page and the campaign hashtag
3. As a business company representative, I'd like to register to the system so I can donate products
4. As a business company representative, I'd like to donate goods to a set of chosen campaigns
   1. I'll provide the amount of product and the value (price in dollars) of each product that I provide to each campaign
5. As a business company representative, I'd like to get the list of social activist and products that I need to ship so I can supply them
   1. The data includes the product id and the user details
6. As a business company representative, I'd like to inform the system that I sent a product to a user so that the system can finish the transaction
7. As a social activist, I'd like to register to the system to earn money and to be able to use the money to buy products
   1. I'll provide my Email, Address, and Phone Number
8. As a social activist, I'd like to tweet about a campaign, so I'll promote it and also earn money
9. As the ProLobby owner, I'd like the system to give a dollar to the social activist for each tweet that promotes a campaign and to each following retweets
   1. The tweet has to have the link and hashtag to the campaign page
10. As a social activist, I'd like to buy products so I can spend the money I earned
11. As the ProLobby owner, I'd like the system to issue a tweet whenever a social activist uses the points to buy a product
    1. The tweet includes the Twitter handle of the social activist and the business company
12. As a social activist, I'd like to know my earning status so I know my balance
13. As a social activist, I'd like to donate a product to my chosen campaign so that I can promote it
    1. I can earn money, buy a product and donate it to a campaign, so the campaign now has more products attached to it
14. As the ProLobby owner, I'd like the system to provide the report about:
    1. Campaigns
    2. Users
    3. Tweets

Non-Functional Requirements

1. Scale - The MVP version can serve a few users concurrently running on the same machine. However, in the future, the system must handle thousands of users without changing the system code
2. High Availability (HA) – The MVP version runs on a single PC and does not provide HA. However, using the same code and services in the cloud should provide HA.
3. Observability – Each operation and each error should issue a log using the standard .NET Microsoft [ILogger](https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.logging.ilogger?view=dotnet-plat-ext-6.0)
4. The Cost of Operation when running in the cloud should be considered. The customer wants to know the cost of operation for a system that handle 20 campaigns, 10 business customers, and 1000 activists
5. Security – No need to secure the system in the MVP
6. Usability - The user interface can be based on a Console or a simple WinForms app for the MVP
7. Hosting - The System runs on the PC, but it uses cloud services hosted locally or .NET facilities

Assumptions & Constraints:

1. Optional local (cloud) services include:
   1. <https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-local> or <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis?view=aspnetcore-6.0>
      1. REST API
   2. <https://github.com/Azure/Azurite> or <https://docs.microsoft.com/en-us/azure/storage/common/storage-use-emulator>
      1. You can use Storage Queue for queuing mechanism. If a better mechanism is required, use <https://www.rabbitmq.com>
   3. <https://docs.microsoft.com/en-us/azure/cosmos-db/local-emulator?tabs=ssl-netstd21>
   4. You can use any SQL DB or MongoDB if required instead of using the CosmosDB emulator
2. Create and use [Twitter developer account](https://developer.twitter.com/en)

The development process and DevOps

The development team has two developers. The process starts by analyzing the requirements and creating the system architecture, i.e., the list of Azure Function apps or Minimal APIs apps responsible for implementing the system, the data structure, algorithms, databases, queues, and any other foundation resource.

After having the system diagram, create a list of development tasks, sorted by priority. Each time a developer finishes a task, it takes the next development task in the list.

Use one of your GitHub accounts and create the development environment, i.e., the GitHub repository. Now, clone the project to each developer machine. Each time a developer starts a new task, the developer creates a task branch. Each time the developer finishes a task, the developer creates a pull request. After the second team member approve the task (by doing a code review), the pull request is merged into the master branch.

Read more about the process [here](https://docs.github.com/en/get-started/quickstart/github-flow).

Testing

Write integration tests by invoking the APIs from a [xUnit.net](https://xunit.net/) project. In addition, you can use [ApprovalTests.net](https://github.com/approvals/ApprovalTests.Net) to simplify the testing process. You may need to mock cloud resources and the Twitter APIs.

DevOps

Use [GitHub Actions](https://github.com/features/actions) to [build and test](https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-net) the code on each pull-request merge.